QUA-KIT DATABASE DESCRIPTION

This document describes the content of the archive I have created to accompany my thesis “Evaluating Symmetry and Order in Urban Design ”. The archive contains a dump of the qua-kit database records to `2016-09-19` to `2018-09-07`. The dump contains such data as student submissions, action records (history of design edits), comments, votes as well as the data specific to the thesis, such as intermediate results of symmetry analysis and final *order scores* of student submissions.

1 Archive contents

`qua-kit-checksums.md5`
This file contains md5 checksums of all files in the archive. You can check the integrity of the data using the following command on a unix-like machine:

```
#$ md5sum -c qua-kit-checksums.md5
qua-kit-db-schema-index.html: OK
qua-kit-db-schema.svg: OK
qua-kit-readme.pdf: OK
qua-kit-readme.tex: OK
qua-kit-thesis-data.sql.xz: OK
```

`qua-kit-readme.pdf`
The archive documentation.

`qua-kit-readme.tex`
The source code of the archive documentation.

`qua-kit-thesis-data.sql.xz`
The dump of the qua-kit database. This is the main file containing all the data.

`qua-kit-db-schema.svg`
Diagram of the database schema. It gives an overview of entities and relationships in the database.

`qua-kit-db-schema-index.html`
Detailed information about the database tables.

2 Extracting data

`qua-kit-thesis-data.sql.xz` is an SQL dump produced by PostgreSQL 10.5 database and archived using LZMA copmression. Although it may be possible to use import the data into other databases, I recommend to install PostgreSQL 10 or later. It is available on Windows, Mac OS, and linux. You can install PostgreSQL on Ubuntu/Debian using the following command:

```
#$ sudo apt-get install postgresql
```

To import the database dump, you need to set up an empty database first; you may need to create a dedicated database user with the rights to create new tables.

The SQL dump file is archived using XZ utils; thus, you need to unpack the data. On Windows, you can use 7-Zip program to unpack the file and then import the dump using the graphical interface of the

PostgreSQL software suite. On a UNIX-like system, both of these operations can be done using a simple one-line command. Assume the PostgreSQL database is named `qua-kit`; the command to import the data is as follows:

```
#$ xzcat qua-kit-thesis-data.sql.xz | psql qua-kit
```

3 Using with qua-kit

You may explore students submissions and comments using the qua-kit interface. To do that, you would need to build and run qua-server application available on `github.com`[1]. The easiest way to do it is to use Haskell stack[2] and `git`. Install these two applications and proceed with the following commands to build qua-kit:

```
#$ git clone https://github.com/achirkin/qua-kit
#$ cd qua-kit/apps/hs/qua-server
#$ stack build
```

...and the following coomand to run qua-kit (in `qua-kit/apps/hs/qua-server` folder):

```
#$ stack exec qua-server
```

Note, `apps/hs/qua-server/config/settingsPostgreSQL.yml` file contains DB connection settings, such as the DB user name and password (defaults `mooc` and `mooc`). If everything works correctly, the site becomes available at `http://localhost:3000/`.

Logging in as a student

You can log in as a student to view the site from their perspective. All students in the database have dummy emails and the same simple password `123`. For example, to log in as a student of the first exercise named Ana Farmer, use email `person277@mail.com` and password `123` in the log in interface. All user emails can be found in table `user`.

4 Database contents

The starting point for exploring the qua-kit database is the schema available in the file called `qua-kit-db-schema.svg`. It shows the complete information on the relationships between qua-kit entities. A more detailed and technical information about data types is presented in the file called `qua-kit-db-schema-index.html`. Some extra information can be gathered by exploring the source code of qua-kit[3].

Chapter 1 of my thesis overviews qua-kit and some of its major components. I recommend to read this chapter prior before working with the database. The qua-kit data can be categorized into six topics: design submissions, design actions, criteria voting, textual reviews and comments, edX integration, and symmetry analysis.

[1] `https://github.com/achirkin/qua-kit`
[2] `https://docs.haskellstack.org/en/stable/README/`
[3] `https://github.com/achirkin/qua-kit/blob/reflex/apps/hs/qua-server/config/models`

4.1 Design submissions

The core of the qua-kit database is the `scenario` table. It represents a single student submission, linked to a `user` (student) and `exercise`.

Table `user` describes any qua-kit user, such as students, experts, or administrators. There is a Haskell type `UserRole`[4] that is mapped onto integers in the DB. For example, all students doing the exercise have role `UR_STUDENT = 1`, administrators have role `UR_ADMIN = 3`.

Table `exercise` describes a design exercise. The `geometry` column is the exercise geometry template (modified `GeoJSON`); the geometry format is described in the qua-view project page[5]. Qua-kit loads this geometry as the initial state of the design submission when a student starts their work. The rest of the table properties define the behavior of the design editor (qua-view).

A user may submit many designs for many exercises. However, only the latest submission of a student is visible. Records in the `scenario` table are immutable: they are never updated or deleted. Instead, I use the `current_scenario` table to keep the id of the last user submission. That is, all student submissions stored as GeoJSON scenario files in column `scenario.geometry`; submission timestamps are stored in `scenario.last_update`; The last submission identifier is stored in `current_scenario.history_scenario_id`. Together with geometry, a student submits a textual description of their work (`scenario.description`) and qua-kit generates a preview image (`scenario.image`, `.png` file).

4.2 Design actions

Design (user) actions are the records reflection activity of a user in the qua-view editor interface; the actions can be used to restore the design process step-by-step. Over the lifetime of the qua-kit site, there were two different ways to log user actions.

Old logging

The old way of logging was user until early 2018 and has majority of the records. This way uses three tables:

- `user_sceanrio_load` – a user has opened the editor; a record contains the timestamp, user and exercise identifiers, and the loaded geometry. Parameter `scale` was used in early versions of qua-view to set up the zoom level of the geometry.
- `user_sceanrio_update` – a custom geometry update specified by the `geometry` record, such as adding new or deleting old geometry.
- `uaer_scenario_action` – an update of the position of a single scenario object (building block). The update is represented by a 4x4 transformation matrix (homogeneous coordinates). Such a matrix potentially can represent any affine transform, but qua-view only allows translations and rotations on XY plane.

A single user session is fully defined by a `user_sceanrio_load` record followed by a series of `user_sceanrio_update` and `uaer_scenario_action` records. Note, `user_sceanrio_load` does not

[4] https://github.com/achirkin/qua-kit/blob/reflex/apps/hs/qua-server/src/Model/CustomTypes.hs#L25
[5] https://github.com/achirkin/qua-view

necessarily refers to an exercise submission, it may refer to an arbitrary qua-view editor session without an `exercise_id` attached.

New logging

The new way of logging allows a larger variety of actions in addition to the object transforms, such as modifying object properties and camera motions. The new logging records are stored in the `qua_view_web_logging` table. The content of a user action is stored in the `action` column as a JSON text. The description of the possible actions can be found on the project page[6].

4.3 Criteria and voting

User submissions are graded by the peer-to-peer grading process based on some design criteria. A record in table `criterion` describes a single design criterion:

- `name` – name of a design criterion, as shown to a student;
- `icon` – an `.svg` icon, must be 24x24 pixels;
- `image` – a `.png` image, must be 200 pixels wide;
- `description` – an `.html`-formatted textual description of a design criterion

The list of criteria used in an exercise is stored in the `exercise_criterion` table.

The results of the voting exercise are stored in the `vote` table; these are the peer-to-peer comparisons of pairs of designs w.r.t. design criteria.

- `voter_id` – a user who voted;
- `criterion_id` – the voting criterion;
- `better_id` – a design selected by the voter;
- `worse_id` – a design NOT selected by the voter;
- `explanation` – an optional textual explanation of the voter decision.

Tables `rating` and `vote_rating` represent the grades computed by qua-kit for the designs and voters respectively. The `value` columns of these tables represent the calculated ratings, the other non-key columns are used by the rating system as described in Chapter 1 of my thesis.

4.4 Text data

There are a few ways qua-kit users can enter data in the text form.

Table `scenario` has column `description`. Users may optionally enter text description alongside with their submission to explain their design ideas and decisions. Even though the text field is optional, students are encouraged to fill it in. Non-empty design description increases chances of other students to positively rate the submission. A large fraction of submissions contains a non-empty description.

Table `review` is the most sensible source of the text data in the database. Reviews are completely optional, and the presence of a review indicates that a student-reviewer is interested in the qua-kit

[6] `https://github.com/achirkin/qua-view/blob/reflex/src/Program/WebLogging.hs#L33`

activities. Reviews always have some sentiment: a user has to upvote or downvote a design with respect to a design criterion; column `comment` is an optional explanation of the user feedback. Approximately a quarter of the reviews in the DB have non-empty comments.

Table `vote` is a part of *compare* exercise; it has optional column `explanation`. Only a small number of votes have non-empty explanations, though the number of votes is large.

There are more text fields in the database; they contain some user or administrator inputs, which may be informative but not suitable for statistical analysis. These are `criterion.description`, `exercise.description`, and `survey`.

4.5 EdX integration

Qua-kit is integrated into the "Future Cities" series of online courses at edX; see Chapter 1 of the thesis for more information. A few table in qua-kit are used to keep the information necessary for authenticating and grading edX students.

The `edx_course` table contains a displayable name and identifiers of an edX course related to a design exercise. A course may contain several entry points into the qua-kit exercise – links in course modules. These are represented by `edx_resource`. Qua-kit keeps a list of parameters known about a resource in `edx_resource_param` table; these parameters are obtained from the request parameters during the oauth authorization. Table `edx_grading` keeps current grades of all students. Table `edx_grading_queue` keeps a list of grades that were updated recently; qua-kit sends updated grades from this list to edX servers once a day.

4.6 Symmetry analysis

The data used in the *order* voting experiment (described in Chapter 3 of the thesis) is stored in table `vote_order`. The table keeps the ids of compared designs, the voter id and the time of the voting. Thus, it is possible to recover individual voting sessions: number of votes and the time span. The content of this table is the input for constructing the crowdsourced order measure.

Table `scenario_analysis` contains intermediate results of the thesis – symmetry voting grids. A symmetry voting grid for a single design submission and symmetry type is stored in table `s_a_image`. The grid itself is a grey-scale `.png` image in column `s_a_image.data`, scaled to range 0-255. To restore the original absolute values, you need to use columns `s_a_image.min` and `s_a_image.max`. Columns `mean` and `var` are present for convenience. Table `scenario_analysis` reference to `s_a_image.data` for every type of symmetry (bilateral reflection, translation, and 2-7-fold rotations). In addition, it keeps a figure-ground image of a design in the same format. The content of the table can be conveniently visualized in qua-kit. To do that, run the qua-server application as described in Section 3, and open page `http://localhost:3000/analysis`.

In the thesis, I calculated the Hamming distance between figure-ground representations of designs to remove duplicate submissions. To aid the removal process, I saved the distances in the `scenario_pixel_dist` table.

Table `scenario_analysis_n_n` is almost the same as `scenario_analysis`, except that it is specialized to serve as the input to the regression model 1 (CNN, Chapter 4). Every scenario in this table is present 10 times, the symmetry analysis is done for rotated versions of a design. All related `s_a_image` records contain small 32×32 images. Finally, some of the records in this table have non-null `symmetry_score`

value (exercise 0 and 1). These are the final symmetry scores computed as described in Chapter 4 using the `glm` model with the regularization parameter value 0.001.

To fill in the analysis tables I developed several Haskell programs in the project folder `haskell/us-qua-kit`, which use `haskell/urban-symmetry` project. Table `scenario_analysis` is filled in by `us-qua-kit`, table `scenario_analysis_n_n` is filled in by `us-qua-kit-nn`, and table `scenario_pixel_dist` is filled in by `us-pixeldist`. The final symmetry scores are set to NULL by default and calculated for two exercises using R script `statistics/regression/regressionNN-ex1.Rmd`.

5 Data anonymization

To protect the student privacy, the user data has been anonymized. I have removed or replaced the following records:

- table `user` column `name` – randomized two-word names;
- table `user` column `eth_user_name` – randomized for users with non-NULL value;
- table `user` column `edx_user_id` – randomized for all users;
- table `user` column `email` – series of dummy (but valid) emails;
- table `user` column `password` – same encrypted value `123` for all users;
- table `qua_view_web_logging` column `ip_address` – set to `NULL` for all records;
- table `user_prop` – removed all records.